DL-HW #02

2015004693_양상헌

(실행환경: Jupyter Notebook)

1. Source Code:

```
#HW for XOR Gate _ 02week
import tensorflow as tf

x_data = [[0, 0], [0, 1], [1, 0], [1, 1]]
y_data = [[0], [1], [1], [0]]
X = tf.placeholder(tf.float32, [None, 2])
Y = tf.placeholder(tf.float32, [None, 1])

W1 = tf.Variable(tf.random_uniform([2, 3], -1.0, 1.0))
b1 = tf.Variable(tf.random_uniform([3],-0.01, 0.01))
layer1 = tf.nn.sigmoid( tf.add(tf.matmul(X,W1), b1) )

W2 = tf.Variable(tf.random_uniform([3, 1], -1.0, 1.0))
b2 = tf.Variable(tf.random_uniform([1], -0.01, 0.01))
output = tf.nn.sigmoid(tf.add(tf.matmul(layer1, W2), b2))

#cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits_v2(logits = output, labels = Y))
cost = - tf.reduce_mean( Y * tf.log(output) + (1 - Y) * tf.log(1 - output) )

opt = tf.train.GradientDescentOptimizer(learning_rate = 0.1)
train_op = opt.minimize(cost)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(10000):
        for x, y in zip(x_data, y_data):
            _, cost_val = sess.run([train_op, cost], feed_dict={X : [x], Y : [y]})
        if step % 1000 == 0:
            print(step, cost_val, sess.run(W1), sess.run(b1), sess.run(W2), sess.run(b2))

    print("\n=======================================================\n")
    print(sess.run(output, feed_dict={X : x_data}))
```

2. Result:

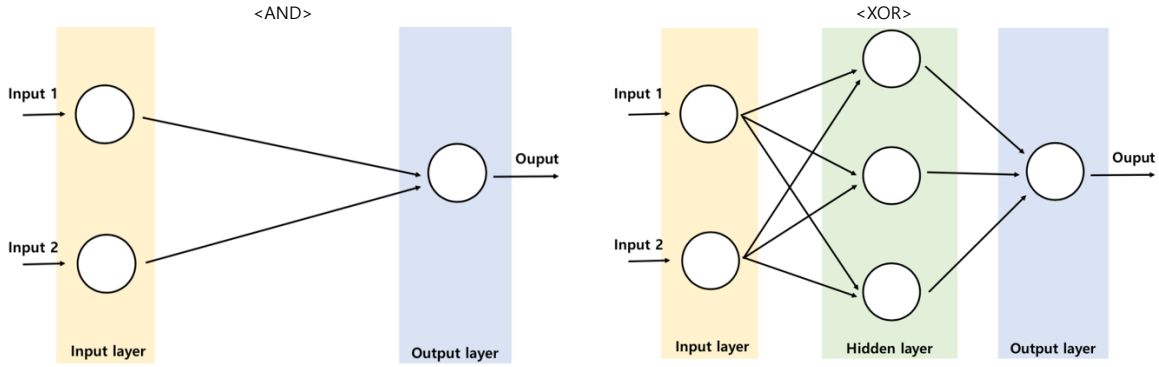
```
0 0.9403354 [[-0.60801554  0.3773584   0.32437983]
 [ 0.10078958  0.57102513  0.35485205]] [-0.00067557 -0.01251432  0.0065244 ] [[ 0.1660611 ]
 [ 0.41543227]
 [-0.0148334 ]] [-0.03577287]
1000 0.7192382 [[-0.9230735   0.23758483  0.04506146]
 [ 0.2421057   0.1038598  -0.19704242]] [-0.48000678 -0.38889754 -0.56624943] [[ 0.41522628]
 [ 0.1758775 ]
 [-0.08806662]] [-0.17785974]
2000 0.21436812 [[-5.0100513   2.8589342  -0.63206667]
 [ 4.671227   -3.2533042  -0.09227131]] [-3.0172992  -1.5215883  -0.68239635] [[ 5.215667]
 [ 4.165433]
 [-0.74408 ]] [-2.0306237]
3000 0.0260585 [[-6.3158712   5.19149   -0.71312374]
 [ 6.0575337  -5.6514907  -0.08458979]] [-3.3673685  -2.8222466  -0.23620632] [[ 8.829483 ]
 [ 8.720717 ]
 [-1.4555598]] [-3.8015828]
4000 0.013196797 [[-6.712534   5.6988463 -0.6525914]
 [ 6.413738  -6.169539  -0.0785156]] [-3.5066717  -3.0788257  -0.07024713] [[10.121106 ]
 [10.016785 ]
 [-1.6500394]] [-4.3126063]
5000 0.008689557 [[-6.9375477   5.97541   -0.6002184 ]
 [ 6.617278   -6.4488783  -0.05328764]] [-3.5901594  -3.214656    0.03756628] [[10.905176 ]
 [10.789796 ]
 [-1.7671895]] [-4.6007233]
6000 0.006408139 [[-7.091786    6.163037   -0.5569662 ]
 [ 6.7592206  -6.6354856  -0.02355512]] [-3.6495037  -3.3058617   0.11905552] [[11.468583 ]
 [11.342015 ]
 [-1.8520048]] [-4.795664]
7000 0.0050392547 [[-7.2075839e+00  6.3040128e+00 -5.2150136e-01]
 [ 6.8679390e+00 -6.7736692e+00  5.9445403e-03]] [-3.6957223  -3.3742409   0.18497896] [[11.908322 ]
 [11.771687 ]
 [-1.9189618]] [-4.9403543]
8000 0.0041322135 [[-7.299539    6.4164057  -0.4924495 ]
 [ 6.955902   -6.882468    0.03358726]] [-3.7336247  -3.428802    0.24040262] [[12.268958 ]
 [12.123163 ]
 [-1.9745497]] [-5.0542054]
9000 0.003490393 [[-7.3753915   6.5095263  -0.4686136 ]
 [ 7.0296464  -6.971512    0.05893476]] [-3.7657738 -3.4742355  0.2882001] [[12.574493 ]
 [12.4206085]
 [-2.02225  ]] [-5.147508]

==================================================

[[0.00308686]
 [0.99730784]
 [0.99732643]
 [0.00300819]]
```

3. Disscussion:

코드의 전체적인 틀은 실습수업시간에 앞서 진행한 AND GATE 구현 코드와 크게 다르지 않다. 다만 AND의 경우 네트워크가 인풋 레이어와 아웃풋레이어 만으로 구성되어도 올바른 결과를 낼 수 있는 반면, XOR의 경우는 그것이 불가능하여 네트워크의 인풋레이어와 아웃풋레이어 사이에 중간 레이어 하나를 추가해 주었다. 이를 그림으로 나타내면 다음과 같다. (사진 출처: 실습자료 MLP.pdf)



위 사진처럼 중간 레이어에 뉴런 3개를 추가하여 주었고, Activation function 같은 경우, 모두 Sigmoid를 사용하였다. 이를 수식으로 표현하면 Sig(W2 * (Sig(W1* X + b1)) + b2) 와 같다. Optimizer는 AND에서와 같은 Gradient Descent Optimizer를 사용했다.

Cost 계산 방식의 경우 AND와는 다른 방식을 사용해야 했기에 강의자료를 참조하여 "cross entropy"를 사용하려 했었다. 하지만 위 코드에서 주석 처리된 부분처럼 사용할 경우 실행을 했을 때 정확한 이유는 모르겠지만 제대로 트레이닝이 이루어지지 않았다. 그리하여 사실상 이와 같은 결과를 내는 "Logistic Regression"의 식을 사용하여 Cost를 계산하였고 이로 인해 트레이닝이 잘 이루어 지게 되었다.

위의 RESULT 사진에서 마지막 4줄이 결과를 나타내는데 각각 순서대로 0, 1, 1, 0 에 가까운 숫자를 출력하는 것을 봤을 때 트레이닝이 잘 이뤄졌다고 볼 수 있다. 처음에는 for 문의 step 횟수를 100, 1000으로 한 상태에서 실행시켰을 때, 거의 대부분 0.3, 0.4, 0.3, 0.4 같은 트레이닝이 잘 되지 않은 결과값을 출력해서 네트워크의 디자인이 잘못되었거나, Cost계산이 잘못 되었다 생각하고 계속 디자인을 조금씩 고쳐봤음에도 별다른 소득을 얻지 못했는데, 단순히 for문의 step 숫자를 10000, 100000 정도로 하고 실행시켰을 때는 위와 같이 0, 1, 1, 0 에 가까운 숫자를 출력했고 트레이닝이 잘 이뤄진 것을 확인할 수 있었다. 결국 Step의 숫자를 충분히 크게 하는 것이 트레이닝에 매우 중요하다는 것을 확인 할 수 있다.